
sinclair
ZX SPECTRUM
Microdrive and
Interface 1 manual

sinclair

ZX INTERFACE 1
and
ZX MICRODRIVE

Designed and prepared by
Cambridge Communication Limited

Note
The names **ZX Microdrive** and **ZX Interface 1** are trademarks of
Sinclair Research Limited

Stanhope Road
Camberley
Surrey
GU15 3PS
Tel: Camberley (0276) 685311

First Edition 1983

Front cover illustration by John Harris of Young Artists

THE ZX INTERFACE 1 . . .

incorporates the three functions of Microdrive controller, local area network and RS232 Interface. Connect it to your Spectrum and you can control up to eight Microdrives, communicate with other computers and drive a wide range of printers.

THE ZX MICRODRIVE . . .

gives you fast access to a large memory. Each Microdrive can hold up to 100K bytes using a single interchangeable cartridge.

THIS MANUAL . . .

introduces the idea of using BASIC as an *operating system* as well as a *programming language.* This way, BASIC can be used to set up communication links, to send and receive data along them and to manipulate files. This adds enormously to its power to perform tasks in which:

- data is stored and retrieved,
- data and programs are sent to peripherals,
- communication links are established with other Spectrum computers or with other types of computer.

The combined use of BASIC as an operating system and a programming language gives it a power and flexibility, and also an ease of use, to be found in few larger computers.

You should read the chapters on the Microdrive even if you have not bought one, since they introduce various concepts (about *channels* and *streams* and their use) which you will need to understand.

This manual should only be read *after* you have familiarised yourself with the ZX Spectrum Introduction booklet and with chapters 1 to 22 of the BASIC programming manual.

Contents

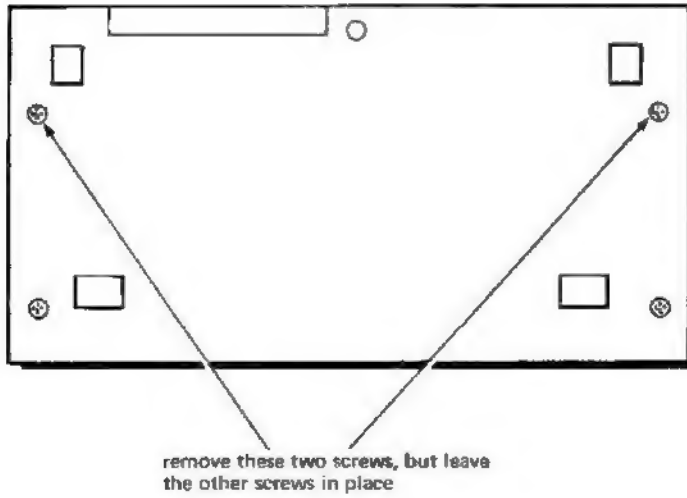
Setting up your ZX Interface 1

Unpacking the ZX Interface 1 you will have found:

- this booklet,
- the Interface itself (with two captive screws on the underside),
- a ribbon cable about 8cms long (for connecting the Interface to a Microdrive),
- a lead with a jack plug at either end (for setting up a network).

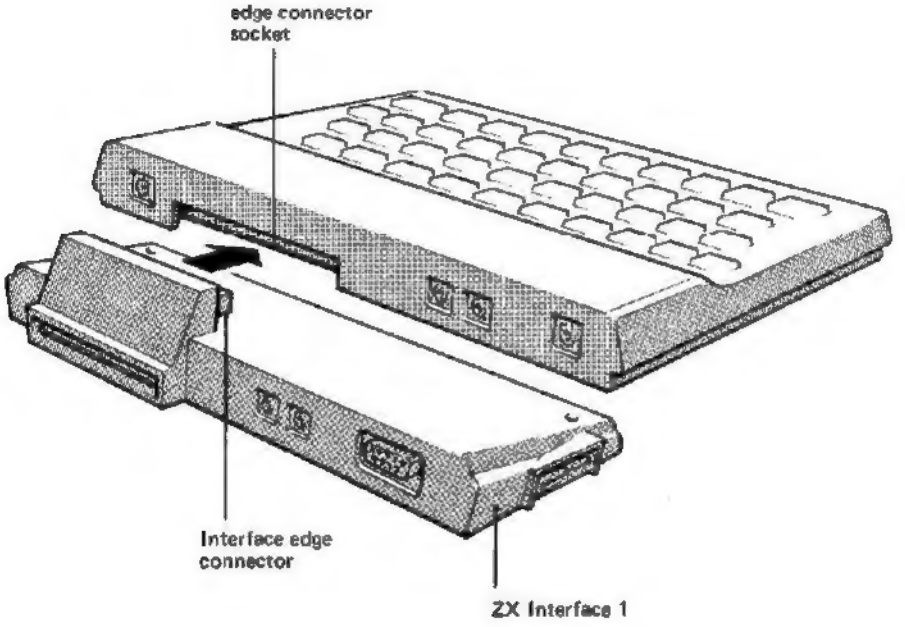
To attach the Interface to your computer you should:

1. Unplug the computer.
2. Disconnect the computer from its peripherals.
3. Using a Posidrive screwdriver remove the two screws on the underside of the computer, as shown in the diagram below. (You will only need these screws again if you later disconnect the Interface from the computer.)

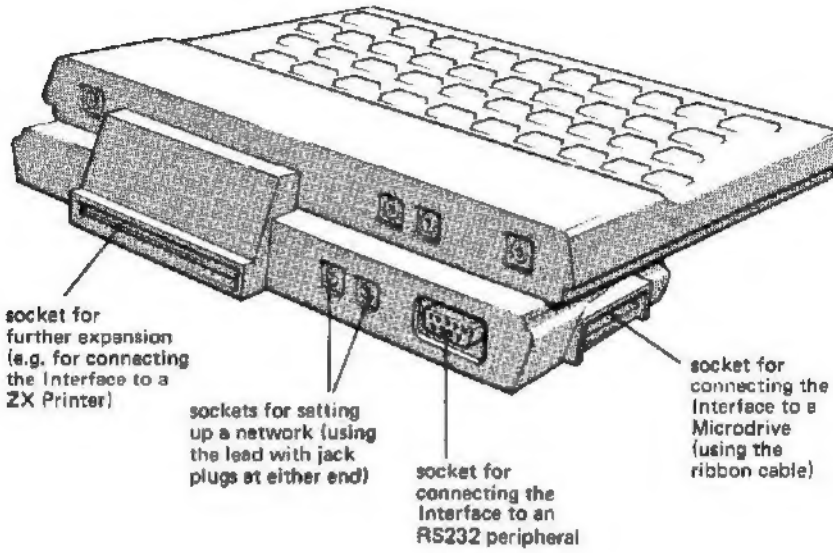


4. Push the Interface edge connector firmly into the socket at the back of the computer.
5. Screw the Interface to the underside of the computer using the two captive screws supplied.

You can now reconnect the computer to its peripherals, and plug it in.



The diagram below shows what the various sockets on the Interface are for.



You can now either:

– set up a Microdrive see page 9,

or

– set up a network see page 29,

or

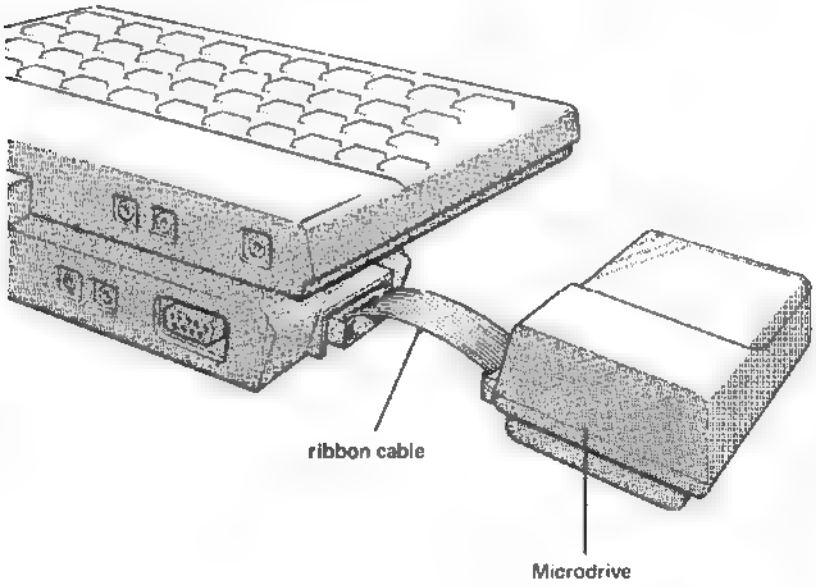
– connect the Interface to an RS232 peripheral see page 35.

Setting up your Microdrive

Unpacking the Microdrive you will have found:

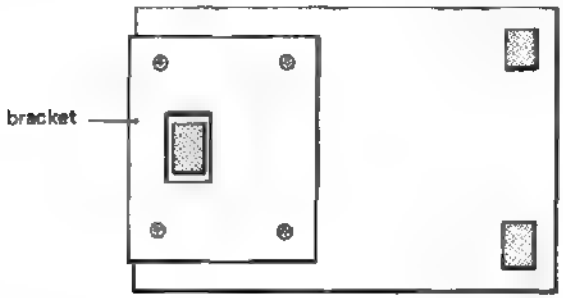
- the Microdrive itself,
- an electrical connector (for connecting this Microdrive to another one),
- a pre-recorded demonstration cartridge.

To attach the Microdrive to the Interface, push one end of the ribbon cable (supplied with the Interface) firmly into the socket on the side of the Interface. Then push the other end of the cable into the socket in the side of the Microdrive.

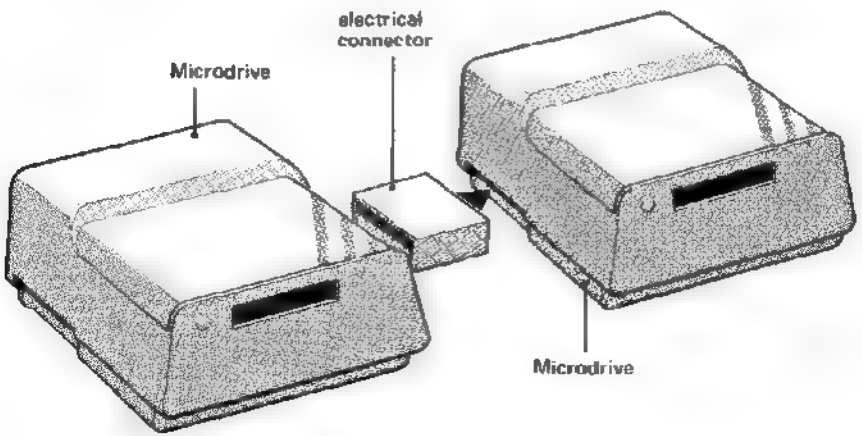


If you have several Microdrives, the second one can be connected to the first as follows.

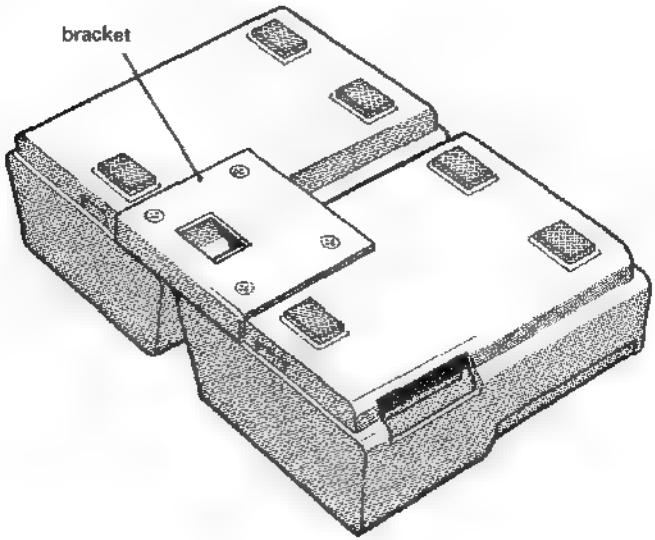
1. Disconnect the Microdrive from the computer.
2. Turn both Microdrives upside down. You will find that each has a bracket screwed to the underside.



3. Unscrew and remove both brackets.
4. Turn the Microdrives the right way up again.
5. Push one of the electrical connectors firmly into the socket on the side of one of your Microdrives (as shown below).
6. Push the second Microdrive up against the first, so that the other end of the electrical connector pushes into the socket on the side of the second Microdrive.



7. Carefully turning the Microdrives over again, screw back one of the two brackets — but this time so that it *links up* the two Microdrives. (The remaining electrical connector, bracket and screws can now be kept for linking up a further Microdrive.)



8. Turn the Microdrives the right way up again, and reconnect the one on the right-hand end to the computer.

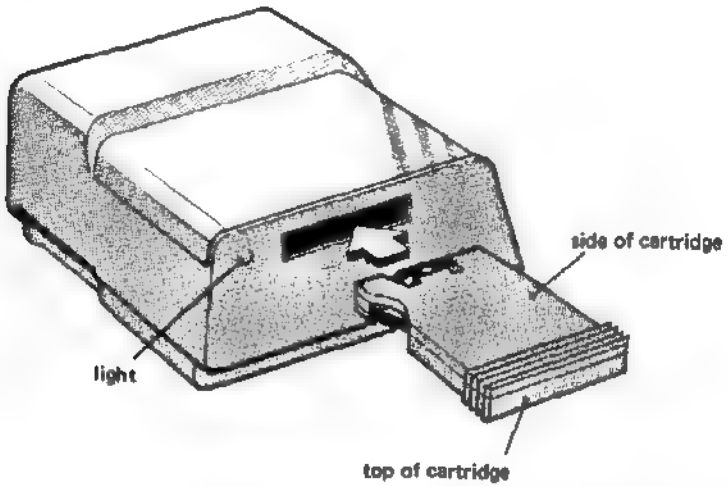
Note that the Microdrive nearest the computer is always known as Microdrive 1, and the next along is Microdrive 2, and so on.

Notice also that on the front of each Microdrive there is a light. This will come on whenever the Microdrive is running.

Handling cartridges

Every cartridge comes in a protective box; and should always be kept in its box when not in use.

When you take a cartridge out of its box (being careful not to touch the tape itself) you will see that it has a label on the top and another on the side.



Push the cartridge firmly into the slot in the front of the Microdrive, making sure that the label on the side faces upwards. This will leave the top of the cartridge projecting by about ½cm. The Microdrive and cartridge are now ready to be used.

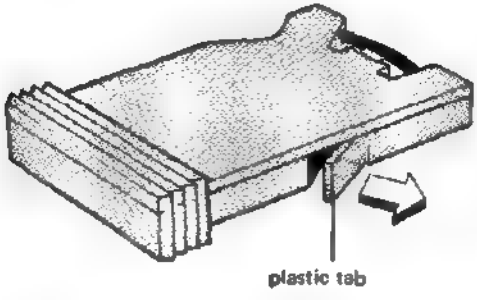
When you have finished using the cartridge, pull it carefully out of the Microdrive, and replace it immediately in its box. But remember . . .

NEVER TAKE THE CARTRIDGE OUT OF THE MICRODRIVE WHILE THE MICRODRIVE LIGHT IS ON.

NEVER SWITCH THE POWER ON OR OFF WHILE A CARTRIDGE IS IN THE MICRODRIVE.

Protecting programs and data stored in cartridges

If you wish to protect the programs and data stored in a cartridge (that is, prevent them from being over-written or erased) then you should snap off the plastic tab from the side of the cartridge, as shown below.



(As with a cassette, you can reverse this effect by sticking some tape over the place where the tab has been removed.)

The cartridge lifespan

Microdrive cartridges will not last forever, and will eventually need to be replaced. The symptom of an ageing cartridge is that the computer will take longer and longer to find a program or file before loading it. So it is a good idea to keep back-up copies of important programs and files either on another cartridge, or on a cassette.

Starting with your Microdrive

Auto-run

Now that you have set up your ZX Interface 1 and your Microdrive, you will want to know what programs are waiting for you on the demonstration cartridge. So, insert the cartridge into the Microdrive (or, if you have several Microdrives, into Microdrive 1), and enter:

NEW

followed by:

RUN (and **ENTER**)

This will automatically load and run the first program on the cartridge. When you have finished watching this program, read on.

The catalogue

To find out what other programs are stored in the demonstration cartridge, enter the CATalogue statement:

CAT 1

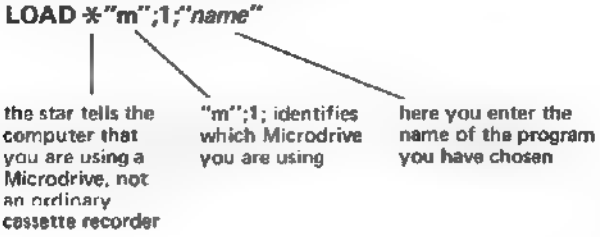
1 identifies the number of the Microdrive you are using

In about seven seconds the television screen will display:

- the name of the cartridge,
- an alphabetical catalogue of the names of all the files stored in the cartridge,
- the amount of memory left in the cartridge (in kilobytes).

Loading programs

The next thing to do is **LOAD** whichever program you would like to see next. So, first choose your program, then enter:



After a short pause, the screen will display the OK message (though not the name of the program). You can now run the program.

Programs and the Microdrive

Saving, verifying, loading and merging programs

In the ZX Spectrum Introduction booklet you found out how to **SAVE** programs on a cassette tape. Saving programs in cartridges is just as easy. Here again is a program called Squares, much like the one in the Introduction booklet:

```
10 REM Squares
20 FOR n=1 TO 10
30 PRINT n,n * n
40 NEXT n
```

To **SAVE** this on a cassette tape you would enter:

```
SAVE "Squares"
```

To **SAVE** it in the cartridge in Microdrive 1, you enter instead:

```
SAVE *"m";1;"Squares"
```

After about six seconds the border will flash, and the program will be saved. Try it and see.

(The names of programs stored in a cartridge, like those of programs stored on a cassette tape, can be up to ten characters long.)

As you would expect, you can now check that the program has been saved by entering:

```
VERIFY *"m";1;"Squares"
```

The screen will then display the OK message.

You can now **LOAD** the program Squares by entering:

```
NEW
```

followed by:

```
LOAD *"m";1;"Squares"
```

then:

```
ENTER
```

Next, to make the program run automatically, try entering:

```
SAVE *"m";1;"Squares 2" LINE 10
```

then:

```
NEW
```

and then:

```
LOAD *"m";1;"Squares 2"
```

Programs can also be merged using a Microdrive. Enter:

```
NEW
```

followed by:

```
100 REM more Squares
110 FOR n=11 TO 20
120 PRINT n,n*n
130 NEXT n
```

And now enter:

```
MERGE *"m";1;"Squares"
```

then:

```
ENTER
```

The program Squares will be added to the listing.

In short, as you have probably already realised, all the syntax used within an ordinary cassette interface (explained in chapter 20 of the BASIC programming manual) applies also to the Microdrive.

Erasing programs

Suppose you have just finished with the program Squares. To erase it, enter:

```
ERASE "m";1;"Squares"
```

(As before "m";1 simply indicates which Microdrive you are using.)

During the **ERASE** statement the border will flash.

If you **BREAK** while saving a program then you will have an unclosed file in the cartridge. You cannot **LOAD** an unclosed file, and an attempt to do so will give the report **file not found.** The **ERASE** Statement can delete an unclosed file, but will take about thirty seconds to do so because the computer checks the cartridge several times to make sure that the file has no end.

Formatting and naming blank cartridges

If you have bought a blank cartridge, before you can use it you must insert it into a Microdrive (say, into Microdrive 1) and enter this:

FORMAT "m";1;"*name*"

"m";1 identifies the Microdrive you are using (in this case, number 1)

enter here whatever name you wish to appear in the cartridge catalogue. It can be up to 10 characters in length

The computer takes about thirty seconds to format a cartridge. During this time the border will first flash, then clear, then flash again, and finally display the OK message. What the computer is doing is identifying any areas that cannot be written to or read from and marking them to be avoided.

Formatting need never be repeated. Note, by the way, that when you format a cartridge, any information it contains will be erased. So, if you have bought several Microdrives, you can **FORMAT** your spare demonstration cartridges and use them as blank cartridges.

Next type:

CAT 1

1 identifies the number of the Microdrive you are using

In a few seconds, the television screen will display the cartridge's name and its capacity. The capacity will vary from cartridge to cartridge, but it should never be less than 85 kilobytes.

Setting up your own auto-run facility

Earlier, you used the auto-run facility on the demonstration cartridge. If you have a program that you often use, you can set up your own auto-run facility, so as to avoid repeatedly having to type the **LOAD** and **RUN** statements. These are the rules to follow:

- the program must be called **run**;
- the cartridge must be used in Microdrive 1;
- the facility must be used either immediately after switching on, or immediately after entering **NEW**.

So, enter the required program, followed by:

10 SAVE *"m";1;"run" LINE *number*

the name **run** must be typed out in full. Do not just press the **RUN** key

enter here the appropriate line number

Now enter:

NEW

followed by:

RUN (and **ENTER**)

enter the keyword, not the file name

Note, however, that the **MERGE** statement does not work with any program saved using the **SAVE*... LINE...** statement. (This is so as to protect such programs.) An attempt to **MERGE** will give the report **'Merge' error.**

Data, channels and streams

As you know, a program is a set of operations which is executed when you type **RUN**. Data, on the other hand, is any collection of letters, numbers or symbols on which a program might operate. An example is the numbers 1 to 10 with their squares.

Data can be sent to, and come from, various parts of a computer system. Such parts are known as *channels*. The channels you can send data to are:

- your screen,
- a ZX Printer,
- a Microdrive file, to store it,
- another ZX Spectrum computer, if both computers are on a network,
- the RS232 Interface and from there, for example, to a modem or printer.

And the channels from which data can come are:

- the keyboard,
- a Microdrive file,
- another ZX Spectrum computer, if both computers are on a network,
- the RS232 Interface, a modem or a terminal.

The routes along which the data flows to and from these channels are known as *streams*. In the Spectrum computer system the number of these streams is fixed at sixteen. They are numbered from 0 to 15, and *stream numbers* are always preceded by the sign #.

Four of these streams come already linked to channels:

stream #0 stream #1	*output* data to the lower part of the television screen and *input* from the keyboard;
stream #2	*outputs* to the upper part of the television screen, but cannot *input*;
stream #3	*outputs* via the ZX Printer; but cannot *input*.

Every statement that produces input or output uses one of these streams automatically. For example, the **PRINT** statement uses stream #2, and the **LPRINT** statement uses stream #3. So, if you enter:

```
PRINT "This is a Spectrum computer"
```

it is, in fact, shorthand for:

```
PRINT #2;"This is a Spectrum computer"
```

Try entering both, and see.

You can, however, make any statement use a different stream by keying # followed by the new stream number. Try entering:

```
LPRINT #2;"This is a Spectrum computer"
```

Instead of being printed by the ZX Printer, this line too appears on the screen.

But as well as using the *established* stream-channels, you can create some of your own. Streams #4 to #15 are free for this purpose; and there are various *channel specifiers* which enable you to indicate which peripheral you require. Some of these are:

"K" for the keyboard

"S" for the screen

"P" for the ZX Printer.

(You will be introduced to the others later.)

Note that K, S and P are all established channels. They require you to use commas (,) as separators in **OPEN #** statements. But with other channels you can use either commas or semi colons (;).

To create your own stream-channels you use the **OPEN #** statement. For example enter:

```
10 OPEN #4,"S"
```

You will have opened stream 4 and linked it to channel S. Now enter:

```
20 PRINT #4; "This is a Spectrum computer"
```

Again, the line will appear on the screen.

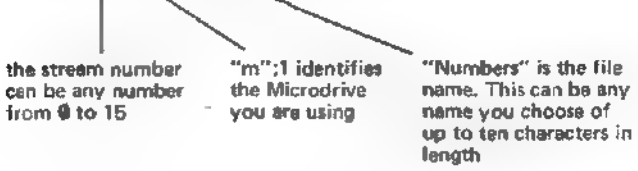
(It is not advisable to **OPEN** to streams **0**, **1** or **2** as the results are unpredictable.)

Data and the Microdrive

Opening and naming a data file

When you store information in a cartridge you keep it in a file. You also give the file a name so that you can find it again later. The statement that opens and names a data file always takes the same form. For example try entering:

```
OPEN #4;"m";1;"Numbers"
```



This statement does two distinct jobs:

- it sets up a new channel: "m";1;"Numbers"
- it attaches this new channel to stream #4.

This will have taken about seven seconds, during which time the computer searches the cartridge for a file called "Numbers". Since there is no file "Numbers", it opens a file for writing. (However, if it *had* found a file "Numbers", it would have opened it for reading. And had it found a *program* "Numbers", it would have given the report **Wrong file type.**)

Entering data

Once you have opened a file you can enter data. Suppose that you want to store the numbers 1 to 10 with their squares. Enter and run this:

```
10 FOR n=1 TO 10
20 PRINT #4,n'n*n
30 NEXT n
```

You might think that all the numbers have now been stored away in the cartridge. But in fact the computer does not automatically transfer anything to the cartridge until a certain amount has built up, which it transfers all at once. This is called *buffering*. A Microdrive buffer is 512 bytes (or characters) long.

To store in the cartridge the data you have entered you must **CLOSE** the file. Until this is done you will be unable to read back from the file.

Closing a file

Closing a file ensures that the file is safely stored in the cartridge. It also closes the channel (in this case "m";1;"Numbers") and leaves the stream (in this case #4) with no channels attached. To **CLOSE** a file you need only **CLOSE** the appropriate stream. So enter:

```
CLOSE #4
```

The border will then flash to show that something is being stored in the cartridge.

(Note that, like the **OPEN** statement, the **CLOSE** statement is followed automatically by #.)

You cannot, by the way, **CLOSE** streams #0, #1, #2 or #3. If you try to do so, streams #0 and #1 will default to channel K; stream #2 will default to channel S; and stream #3 will default to channel P (see page 22).

Reading back from a file

To read back from the file "Numbers", run this:

```
10 OPEN #4;"m";1;"Numbers"
20 FOR b=1 TO 10
30 INPUT #4;m;n
40 PRINT "The Square of ";m;" is ";n
50 NEXT b
60 CLOSE #4
RUN
```

leave spaces here

(Note at this point that because the file "Numbers" already exists, the channel "m";1;"Numbers" is opened for input, and trying to output it would give an error.)

You can also use **INKEY$** to read back from a file (it always gives the next character in the file). Try this program:

```
10 OPEN #11;"m";1;"listing"
20 LIST #11
30 CLOSE #11
40 OPEN #12;"m";1;"listing"
50 PRINT INKEY$ #12;
60 GO TO 50
```

This will finish with the **End of file** report.

Notes on PRINT and INPUT

Because the **PRINT** and **INPUT** statements are designed mainly for use with the screen and keyboard, you must take care when using them with files.

separators

The **PRINT** statement uses three forms of separator:

- the ; (semi colon) prints nothing,
- the , (comma) takes you to the start of the next half line,
- the ' (apostrophe) gives a new line (the **ENTER** code).

The **INPUT** statement always expects you to type **ENTER** after a number or a string. So, when you are printing to any file from which you expect to **INPUT**, you must either:

- print the items singly, e.g.

```
10 PRINT #4;2
20 PRINT #4;3
```

or

- separate them with an apostrophe, e.g.

```
10 PRINT #4;2'3
```

You must also take care when using separators in an **INPUT** statement. As you know, **INPUT** can print to the bottom half of the screen anything that you can put in a **PRINT** statement. But when you **INPUT** from a file, the file is only open for reading. So, if you include anything that would be printed when using the screen, you will get the error report **Writing to a 'read' file.** This means that items in the **INPUT** statement should be separated with a semi colon, e.g.

```
10 INPUT #4;a;b
```

Be careful also when you **INPUT** a string containing " (quotes), because the **INPUT** will think that the " is the end of the string. The way round this is to replace, for example:

```
10 INPUT #4;a$
```

with

```
10 INPUT #4; LINE a$
```

changing streams

PRINT statements may also contain information for several streams at a time. The following program will print "one" on the screen; "two" to a Microdrive file called "digits" in Microdrive 1; "three" to station 1 on a network (see chapter 7); and "four" to the next line on the screen.

```
10 OPEN #4;"m";1;"digits"
20 OPEN #5;"n";1
30 PRINT "one"; #4;"two";#5;"three" '#2"four"
40 CLOSE #4
50 CLOSE #5
```

setting colours

After using a channel other than the screen, you may find that the **PAPER** and **INK** statements have no effect. To get round this, enter:

```
PRINT;
```

before setting **PAPER** and **INK**.

For more about **PRINT** and **INPUT** see chapter 15 of the Basic Programming manual.

Reading the file catalogue

As you establish files in a cartridge they are automatically catalogued. So, when you want to find out what files a cartridge contains, you need only insert the cartridge into a Microdrive and enter the CATalogue statement. For example, enter:

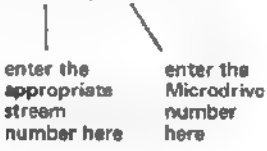
```
CAT 1
```

The television screen will display:

- the cartridge name,
- the file names,
- the amount of memory left in the cartridge, (in kilobytes).

You can also send the output of **CAT** to a stream by entering:

CAT # *number,number*



This enables you to send the catalogue to a printer, or to a file, so that a program can use it.

Protecting a file

If you do not want a file name to appear on the catalogue, you can protect it by giving it a name beginning with the character whose **CODE** is 0. Enter this:

```
10 OPEN #4,"m";1;CHR$0+"Results"
20 FOR n=1 TO 15
30 PRINT #4;n,n*n
40 NEXT n
50 CLOSE #4
```

Now enter:

```
CAT 1
```

The file name will not appear. So, whenever you create a protected file, remember to make a note of its name somewhere, in case you forget it later!

Extending a file

Suppose that you want to extend the file "Numbers" to include the squares of the numbers 1 to 20 instead of only 1 to 10. You cannot reopen a file for writing, so you will have to:

- make a new version with a different name;
- transfer the old file to the new version;
- add the extra data;
- delete the old file.

Here is how to do it.

First, RUN this program:

```
10 OPEN #4;"m";1;"Numbers": REM for reading
20 OPEN #5;"m";1;"Numbers 1": REM for writing
30 FOR f=1 TO 10
40 INPUT #4;m;n
50 PRINT #5;m'n
60 NEXT f
70 FOR n=11 TO 20
80 PRINT #5;n'n*n
90 NEXT n
100 CLOSE #4:CLOSE #5
```

Now, to check that you have got two files, "Numbers" and "Numbers 1", enter:

```
CAT 1
```

Next, to delete the old file, enter:

```
ERASE "m";1;"Numbers"
```

To check that it has been deleted, enter:

```
CAT 1
```

The file name "Numbers" will have disappeared from the catalogue, and your new file, "Numbers 1" now contains the numbers from 1 to 20.

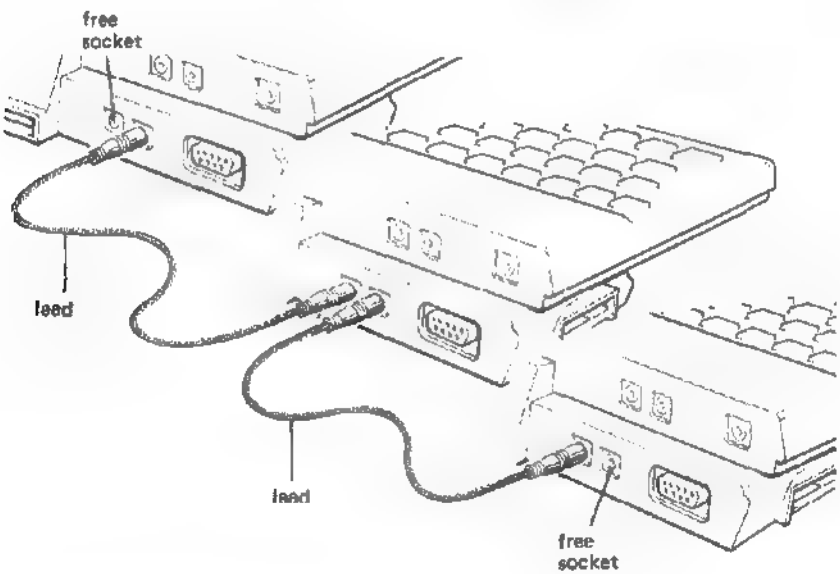
The Local Area Network

Setting up a network

The local area network, or *net*, enables you and your friends to play computer games together, and to send each other programs and data. This means that only one of you need ever type in a program. A net is especially useful, too, if only one of you has a Microdrive.

Using the lead supplied with each Interface you can link up as few as two and as many as sixty-four Spectrum computers, as shown below.

Note, however, that you and your friends should not form a *loop* of computers: the computers at each end of the net should never be connected to each other. Each should be left with one net socket free.



NEVER SWITCH ON OR OFF A SPECTRUM WHICH IS ON A NET WHILE COMMUNICATION IS IN PROGRESS. You may, however, have a switched off Spectrum on the net; and you may also switch on or off Spectrums which are on a net *provided that communication is not in progress.*

When you have set up a net, each computer (or *station*) should be given a different identification number. So, first decide with your friends what number each station is to have, and then each of you should enter:

FORMAT "n";*number*

enter here the station number you have chosen

Note, incidentally, that if there are only two computers on a net, both can use the same station number. And, since both computers, when switched on, become station 1 automatically, there is no need to use the **FORMAT** statement at all.

Programs and the network

Let us suppose that you have a friend called Jack and that both of your computers are on a net. Your station number is, let us say, 1, and Jack's is 2.

Suppose that you want to send Jack this program:

```
10 REM roots
20 FOR n=1 TO 10
30 PRINT n,SQRn
40 NEXT n
```

Enter the program, followed by:

SAVE*"n";2

(Notice that the net does not use program names.)

Meanwhile, Jack should enter:

FORMAT "n";2

followed by:

LOAD*"n";1

Jack will now have a copy of the program. Notice, though, that while the computer is waiting to **SAVE** or **LOAD** a program over the net, the border of the screen goes black until the program is sent. Your computer will not send until Jack's is ready, and Jack's will wait until something is sent. Try entering your **SAVE** line before Jack enters his **LOAD** line, and vice versa.

To verify that Jack now has a copy of your program, he should enter:

```
VERIFY*"n";1
```

while you repeat the sending of the program by entering:

```
SAVE*"n";2
```

SAVE is, in fact, the only statement that sends programs over the net. The **LOAD, VERIFY** and **MERGE** statements are all ways of receiving programs.

The net game in Appendix 1 is a good example of how to use programs with a network.

Data and the network

Suppose that you now want to send Jack some data. The statement **OPEN #4;"n";2** opens a channel to station 2 on the net ("n"), and attaches stream **#4**, to it, so that when you output along stream **#4**, your message will be put on the net with a note saying that it comes from you.

(Conversely, were you to enter **INPUT #4;"n";2** your computer would wait for information addressed to you from Jack.)

Now enter this program:

```
10 OPEN #4;"n";2: REM for output
20 INPUT a$: PRINT #4;a$
30 CLOSE #4
40 OPEN #4;"n";2: REM for input
■ INPUT #4;b$: PRINT b$
60 CLOSE #4
70 GO TO 10
```

Then enter:

```
SAVE *"n";2
```

Now ask Jack to enter:

```
FORMAT "n";2
```

then:

```
LOAD *"n";1
```

You should now run your program, and Jack should edit lines **1Ø** and **4Ø** of the program to make them refer to station 1 not station 2. He should then enter:

```
GO TO 4Ø
```

You are now ready to start a conversation. But before you do so there are three things you should know.

- Your output along stream **#4** is *buffered*: that is, it is not automatically put on the net until a certain amount of output has accumulated. So whenever you finish outputting you should **CLOSE** the stream. This will send the buffer even if it is not full. (The buffers are 255 bytes, or characters, long.)
- Your output is marked as coming specifically from *you*, so that if Jack is inputting or waiting for output from a different net channel, your message will be ignored. If your message *has* been ignored your screen will not display the **OK** message, and the border of the screen will go black, until Jack asks to receive the message.
- Though the **INPUT** statement simply waits for something to be sent, the **INKEY$** statement can be used to read the net. It will then return with the first byte of anything that either has been sent, or is waiting to be sent. (Otherwise it will return the empty string.) This is known as polling. (**INKEY$**, in fact, works the same way with the net as it does with the keyboard.)

The program below will print anything being sent to it by station 1:

```
1Ø OPEN #8;"n";1
2Ø PRINT INKEY$ #8;
3Ø GO TO 2Ø
```

(For more on **INKEY$** see chapter 18 of the BASIC programming manual.)

Broadcasting

There is also a special net channel called *broadcast* whose channel specifier is "n";0. When you input from this channel you will pick up any message that is being broadcast. And when you output, your message can be read by anyone who is inputting from channel "n";0.

This could, for example, be very useful at school if every pupil in a class has a Spectrum computer, but only the teacher has a Microdrive.

Suppose the teacher wished to broadcast a program. First, the pupils should enter:

```
LOAD *"n";0
```

This will leave the pupils 'waiting' to receive the program. The teacher should then save the program by entering:

```
SAVE *"n";0
```

As you may have guessed, broadcasts (unlike private messages) are sent at once, and do not wait for other computers to be ready to receive them.

Note, by the way, that when you send a broadcast, the computer does not let you know if anyone has received it.

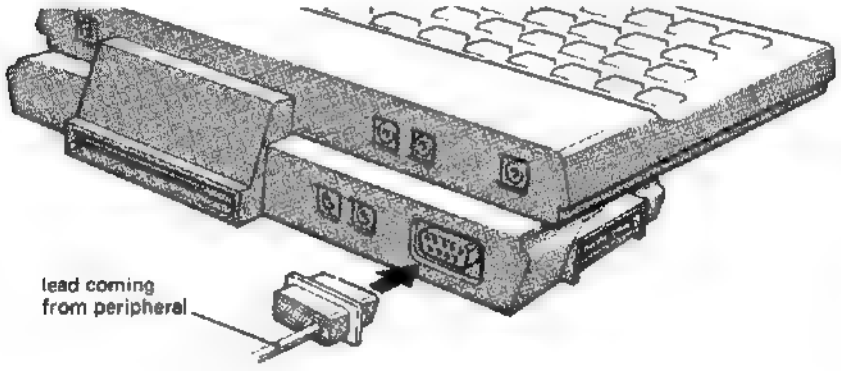
Note also that **INKEY$** will not poll the broadcast channel. Like **INPUT**, it will simply wait for something to be sent.

Using the RS232 Interface

Connecting peripherals to the RS232 Interface

As you know, the Spectrum character set comprises both single *symbols* (letters, digits, etc.) and compound *tokens* (keywords, function names, etc.). All these characters can be sent and received by the RS232 Interface to and from any compatible serial device; for example, a printer, a modem or another RS232 Interface connected to a different kind of computer.

To connect any of these peripherals to the RS232, you should use a lead with a 9-way D-plug on one end and an appropriate plug for your peripheral on the other. You should insert the D-plug into the back of the RS232, as shown below. (For wiring details, see Appendix 4, page 49.)



Next, before you can use the RS232 you will need to adjust some of the settings on the peripheral. These may include:

- the auto line feed, which should normally be switched to off. (The Spectrum outputs a carriage return and a line feed on a "t" channel but a carriage return only on a "b" channel. These t and b channels are explained below.)
- the parity on/off, which should be set to off.
- the number of bits, since the Spectrum puts out eight bits.
- the number of stop bits, since the Spectrum puts out one stop bit.

– the *baud rate* (i.e. the number of bits per second). The Spectrum can operate at all the standard baud rates, i.e.:

  50, 110, 300, 600, 1200, 2400, 4800, 9600 and 19,200.

  You should set your peripheral to use the highest baud rate of which it is capable. (You will see later how to set the Spectrum to use the same baud rate.)

For an explanation of how to make these adjustments, consult the instruction book supplied with your peripheral.

t and b channels

The RS232 uses two different channels: the **t** channel and the **b** channel.

The t channel

The **t** (or text) channel is normally used to send listings. Channel **t** has the following effect on the character set:

characters	
0–31	(the control codes) are not sent, except for 13 (carriage return) which is sent as 13 followed by 10 (carriage return and line feed).
32–127	(the ASCII codes) are sent as normal.
128–164	*(the graphic characters) are not sent. The message ? (code 63) is* sent instead.
165–255	(the tokens) are expanded into ASCII codes 32–127.

On **INPUT** and **INKEY$** the **t** channel expects 7 bit characters, so it ignores the 8th bit if there is one.

To use the t channel, first you must tell the computer which baud rate to use. So enter:

10 FORMAT "t";*baud rate*

enter here the baud rate you have set your peripheral to use

Now to **OPEN** a stream to it, enter:

```
20 OPEN #3;"t"
30 LLIST
```

The border of the screen will flash and the listing will be sent to the peripheral. (Notice that **LLIST** is shorthand for **LIST #3**.) Now enter:

```
LPRINT "This is a short message."
```

This message, too, will be sent to the device.

If your ZX Interface is connected to a terminal or an RS232 device capable of sending, then you can **INPUT** from the terminal or device. Enter this:

```
10 FORMAT "t";baud rate
20 OPEN #4;"t"
30 PRINT INKEY$ #4;
40 GO TO 30
```

enter here the baud rate you have set your peripheral to use

Now, whatever you type on the terminal or device will appear on your screen.

The b channel

The **b** (or *binary*) channel sends the full 8 bit codes used by the Spectrum, and allows you to send control codes to printers etc.

With **INPUT** and **INKEY$** the **b** channel expects 8 bit characters.

SAVE and **LOAD** will only work with the **b** channel.

If you are using modems to connect your computer with another Spectrum over a 'phone line; or if you are storing your programs on a different kind of computer which also has an RS232 Interface, you may well want to **SAVE** and **LOAD** over the RS232 Interface. To do this, enter:

```
FORMAT "b";baud rate
```

enter here the baud rate you have set your peripheral to use

Now, for example, enter:

```
10 REM figures
20 FOR n=1 TO 10
30 PRINT n,n*RND
40 NEXT n
```

followed by:

```
SAVE *"b"
```

If this program is being sent via a modem, someone at the other end should enter:

```
LOAD *"b"
```

The usual extensions are also possible, such as:

```
SAVE *"b";SCREEN $
```

and:

```
SAVE *"b";LINE number
```

Sending control codes

Many printers allow you to send instructions for such operations as double width printing. To send these you should use the b channel. However, with the b channel, the carriage return does not include a line feed. So you will have either to change the settings on your printer, or print the line feed code: 10.

For example, suppose that the control code for double width printing is 14. Enter:

```
10 OPEN #4;"b"
20 PRINT #4;"Normal width"
30 PRINT #4;CHR$ 14;"Double width"
40 CLOSE #4
```

(For the correct control codes to use, refer to your printer's instruction book.)

You can have both b and t channels open at once. Try entering this:

```
10 OPEN #5;"b"
20 OPEN #6;"t"
30 PRINT #5;CHR$ 14;
40 LIST #6
50 CLOSE #5: CLOSE #6
```

This would give you a double width listing.

The MOVE statement

So far, you have only been able to move data from a program to a channel or vice versa. The **MOVE** statement, however, enables you to move data from one channel to another. For example, to move data from the keyboard to the screen, enter:

```
10 MOVE #1 TO #2
```

then:

```
RUN
```

Anything you type on the keyboard will now appear on the screen. However, you will discover that when you press **BREAK** this only prints a space on the screen. To escape from this trap, press **ENTER** until the print position reaches the bottom of the screen. Then, when the computer asks **scroll?** you should press **BREAK**. (You should, by the way, avoid moving data from the keyboard to any other stream since you may be unable to **BREAK** out of such a mode.)

Using the **MOVE** statement you can also examine files stored in cartridges. For example, set up the file "Numbers" (see page 23) and then, to examine its contents, enter:

```
10 MOVE "m";1;"Numbers" TO #2
```

(Note that you need not **OPEN** or **CLOSE** the file yourself. **MOVE** does this.)

Similarly, to make a copy of the file "Numbers" enter:

```
10 MOVE "m";1;"Numbers" TO "m";1;"Numbers 2"
```

Here, **MOVE** opens a stream for reading from the existing file ("Numbers") and another for writing to the new file ("Numbers 2"). Next, it reads the data in "Numbers" and writes it out in "Numbers 2". Then it closes both streams.

MOVE will work with stream numbers (such as #4), and with channel specifiers (such as **"m";1;"Numbers"**). Note, however, that the established streams, #1 to #3, may *not* be specified by the channel specifiers **K**, **S** or **P**.

If you have a second Microdrive, you can use the **MOVE** statement to make back-up copies of data in another cartridge. Enter:

```
10 MOVE "m";1;"Numbers" TO "m";2;"Numbers 2"
```

(Note that **MOVE** only works with data files. If you want a back-up copy of a program, you must **LOAD** the program, and then **SAVE** it.)

It is sensible to make back-up copies of any data or programs you want to keep.

The **MOVE** statement can also send files to a printer. So if you have a ZX Printer, enter:

```
10 MOVE "m";1;"Numbers" TO #3
```

The Printer Server program

This program allows one Spectrum on a net to control an RS232 printer. The printer can then be used by all the other computers on the net. This is useful, for example, if a group of people using Spectrums have only one high quality printer among them which they wish to share. The program also shows a powerful use of the **MOVE** statement.

The Printer Server computer must always be station 64, and must always make contact with station 62 (which is a contact-establishing station). So the sender temporarily uses station 62, and sends it his real station number from which it then moves a file to the t channel. To set up a Printer Server station use this program:

```
10 FORMAT "n";64
20 OPEN #4;"n";62: INPUT #4;a$: CLOSE #4
30 MOVE "n";CODE a$ TO "t"
40 OPEN #4;"b": PRINT #4;CHR$ 12: CLOSE #4: RUN
```

(Line 40 sends a form feed.)

The program below is the one then used by the sender. First, the sender's station is temporarily set to station 62. Then, the sender's real station number is sent. Next the sender's station sets itself back to its real number. Last, line 60 sends whatever data is to be printed (in this case, the listing).

```
10 LET station=number
```

enter here the sender's real station number

```
20 FORMAT "n";62
30 OPEN #4;"n";64: PRINT #4;CHR$ station: CLOSE #4
40 FORMAT "n";station
50 OPEN #4;"n";64
60 LIST #4
70 CLOSE #4
```

The net game

There is a copy of this game on the demonstration cartridge supplied with the Microdrive. Its file name is **"net game"**. The program gives a good example of how the net can be used. Parts of it might usefully be included in programs of your own.

The game

To play this game, you and your opponent must each think of a number between 1 and 100. The winner of the game is the one who guesses his opponent's number first. At each guess you make, your computer will tell you how close you are getting.

The program

The subroutine at line 500 decides who is user 1 and who is user 2. This is so that, when you exchange guesses, one of you uses the subroutine at line 1100, and the other uses the subroutine at line 1200; and thus user 1 sends first and user 2 receives first.

Your computer decides who is user 1 by sending your opponent's computer the message "1", and then listening. If it receives back a "1", this means that your opponent's computer was switched on after yours. Your computer therefore sends a "2" to your opponent's computer, and makes itself user 1. (If, on the other hand, your computer receives back a "2", this means that your opponent's computer was already switched on and listening when your computer sent the "1". Your computer will therefore make itself user 2.)

If both computers start at the same time they will collide and it will be necessary to **BREAK** and start again.

The body of the program is involved with exchanging names, inputting the secret number (which is not sent) and then comparing guesses. First, guesses are sent, and then the replies.

Lines 190 onwards detect a win, signal it appropriately, and then offer another game.

```
10  GO SUB 500
20  PRINT; : BORDER 1 : PAPER 1 : INK 7 : CLS
30  PRINT "  Number guessing game" ' ' "First enter your secret number,
    then guess your opponent's"
40  INPUT "What is your name?";a$
50  PRINT ' ' "Hello";a$
60  GO SUB 1000+100*user
70  PRINT "You are playing";b$
75  PRINT 'a$, b$
80  INPUT "Think of a number (1 to 100)";a
90  IF a<1 OR a>100 OR a <> INT a THEN GO TO 80
130 INPUT "Make a guess";b
140 LET a$=STR$b :  GO SUB 1000+100 user
150 LET c=ABS (a-VAL b$)
160 IF c=0 THEN LET a$="Right" : GO TO 170
161 IF c<4 THEN LET a$="Very very close" : GO TO 170
162 IF c<10 THEN LET a$="Very close" : GO TO 170
163 IF c<20 THEN LET a$="close" : GO TO 170
164 IF c<40 THEN LET a$="fairly close" : GO TO 170
165 IF c<60 THEN LET a$="not very close" : GO TO 170
166 LET a$="nowhere near"
170 GO SUB 1000+100*user
180 PRINT b$,a$
190 IF c=0 OR b$="Right" THEN GO TO 210
200 GO TO 130
210 IF b$="Right" THEN PRINT FLASH 1;"Victory" : FOR n=0 TO 7 :
    BORDER n: BEEP .1,n : BEEP .1,n+16 : NEXT n : GO TO 230
220 PRINT ,"Defeat" : FOR n=7 TO 0 STEP -1 : BORDER n :
    BEEP .2,n : NEXT n
230 BORDER 1 : INPUT "Another game? (y/n)";a$
240 IF a$="y" THEN RUN 20
499 STOP
500 OPEN #4;"n";0
510 PRINT #4;"1"
520 CLOSE #4
530 OPEN #4;"n";0
540 INPUT #4;a$
545 CLOSE #4
550 LET a=4 : IF a$="1" THEN OPEN #a;"n";0: PAUSE 5 : PRINT #4;
    "2" : LET user=1
560 IF a$="2" THEN LET user=2
```

```
 570  CLOSE #4
 580  FORMAT "n";user : RETURN
1100  OPEN #4;"n";3—user
1110  PRINT #4;a$
1120  CLOSE #4
1130  OPEN #4;"n";3—user
1140  INPUT #4;b$
1150  CLOSE #4
1160  RETURN
1200  OPEN #4;"n";3—user
1210  INPUT #4;b$
1220  CLOSE #4
1230  OPEN#4;"n";3—user
1240  PRINT #4;a$
1250  CLOSE #4
1260  RETURN
```

System variables

In addition to the system variables given in Chapter 25 of the Spectrum BASIC programming manual, the Microdrive, local area network and RS232 software use the system variables below.

Notes	*Address*	*Name*	*Contents*
X1	23734	FLAGS 3	Flags
X2	23735	VECTOR	Address used to extend the BASIC interpreter
X10	23737	SBRT	ROM paging subroutine
2	23747	BAUD	Two byte number determining the baud rate calculated as follows: BAUD=(3500000/(26*baud rate))−2 You can use this to set up non-standard baud rates
1	23749	NTSTAT	Own network station number
1	23750	IOBORD	Border colour used during I/O You can **POKE** any colour you want
N2	23751	SER_FL	2 byte workspace used by RS232
N2	23753	SECTOR	2 byte workspace used by Microdrive
N2	23755	CHADD_	Temporary store for CH_ADD
1	23757	NTRESP	Store for network response code
1	23758	NTDEST	Beginning of network buffer contains destination station number 0–64
1	23759	NTSRCE	Source station number
X2	23760	NTNUMB	Network block number 0–65535
N1	23762	NTTYPE	Header type code
X1	23763	NTLEN	Data block length 0–255
N1	23764	NTDCS	Data block checksum
N1	23765	NTHCS	Header block checksum
N2	23766	D_STR1	Start of 8 byte file specifier 2 byte drive number 1–8
N1	23768	S_STR1	Stream number 1–15
N1	23769	L_STR1	Device type . . . "m", "n", "t" or "b"
N2	23770	N_STR1	Length of filename
N2	23772		Start of filename
N8	23774	D_STR2	Second 8 byte file specifier used by **MOVE** and **LOAD** commands
N1	23782	HD_00	Start of workspace for **SAVE, LOAD, VERIFY** and **MERGE** data type code

Notes	*Address*	*Name*	*Contents*
N2	23783	HD_ØB	Length of data Ø–65535
N2	23785	HD_ØD	Start of data Ø–65535
N2	23787	HD_ØF	Program length Ø–65535
N2	23789	HD_11	Line number
1	23791	COPIES	Number of copies made by **SAVE**
	23792	Start of Microdrive MAPs or CHANS	

WARNINGS

1. Opening a stream to the Microdrive or net requires a certain amount of free memory with which to create a channel. A Microdrive channel is 595 bytes, and a net channel is 276 bytes. These channels will be created either by **OPEN#**, **MOVE** or by **SAVE/LOAD/VERIFY/MERGE**. This means that an existing program with insufficient room below RAMTOP will give the report **Out of memory** to any of these operations.

2. Another effect of the creation of these buffers is to move machine code stored in a REM statement. This may create problems. So always put machine code programs above RAMTOP.

3. It is inadvisable to **BREAK** during a Microdrive write operation (one during which the border is flashing), since you may end up with an unclosed file. **ERASE** will remove unclosed files, but will take about thirty seconds to do so, as the computer checks the cartridge several times to make sure that the file has no end.

Microdrive channel

Every time a file is opened an area called a CHANNEL is created in the area designated CHANS in the BASIC programming manual. This area is usually addressed by the IX register in the software. The channel has a length of 595 bytes, and contains the 512 byte buffer.

The contents of the channel are as follows:

0		Address 8
2		Address 8
4		'M'
5		Address of output subroutine in ROM
7		Address of input routine in ROM
9		Address 595
11	CHBYTE	Current byte counter indicates the next byte to be added or removed from the data area in the range 0–512 inclusive
13	CHREC	Record number. Indicates the position of the record in a file range 0–255
14	CHNAME	10 byte filename with trailing spaces
24	CHFLAG	Flag byte bit 0 set . . . open for write clear, open for read bits 1–7 . . . unused
25	CHDRIV	Drive number 0–7
26	CHMAP	Address of the MAP for this Microdrive
28	CHMAP	12 bytes of header preamble . . . marks the start of the header workspace
40	HDFLAG	Flag byte bit 0 set bits 1–7 . . . unused
41	HDNUMB	Sector number in range 0–255
42		Unused
44	HDNAME	Cartridge name and trailing spaces
54	HDCHK	Header checksum
55		12 bytes of data block preamble . . . marks the start of the data workspace
67	RECFLG	Flag byte bit 0 = 0 bit 2=not a **PRINT** file bits 3–7 unused
68	RECNUM	Number of this record in the range 0–255

69	RECLEN	Number of bytes of data in this record 0–512
71	RECNAM	Filename with trailing spaces
81	DESCHK	Checksum of the preceding 14 bytes
82	CHDATA	512 bytes of data
594	DCHK	Checksum of the preceding 512 bytes

MAP

For every Microdrive containing an opened file there is an area called a MAP created in the area called "Microdrive maps" in the BASIC programming manual. The MAP contains 32 bytes. Each bit corresponds to a sector on the corresponding Microdrive. If that sector contains data or if it is unusable then the bit is set. The bits are numbered as follows: bit 0 byte 0 = sector 0, bit 1 byte 0 =sector 1, bit 0 byte 1 = sector 8, and so on.

Network channel

When a stream is opened to the network, a channel is created in the area designated CHANS in the BASIC programming manual. This area is usually addressed by the IX register in the software. The channel has a length of 276 bytes, and contains the 255 byte buffer.

The contents of the channel are described as follows:

0		Address 8
2		Address 8
4		"N"
5		Address of output subroutine in ROM
7		Address of input subroutine in ROM
9		Address 276
11	NCIRIS	The destination station number
12	NCSELF	This SPECTRUM's station number
13	NCNUMB	The block number
15	NCTYPE	*The packet type code . . . 0 data, 1EOF*
16	NCOBL	Number of bytes in the data block
17	NCDCS	The data checksum
18	NCHCS	The header checksum
19	NCCUR	The position of the last character taken from the buffer
20	NCIBL	The number of bytes in the input buffer
21	NCB	A 255 byte data buffer

RS232 connections

The RS232 socket is wired as follows:

1. No connection
2. TX data (input)
3. RX data (output)
4. DTR (input) this should be high when ready
5. CTS (output) this should be high when ready
6. n.c.
7. Ground (pull down)
8. n.c.
9. +9v (pull up)

An RS232 cable is available from Sinclair Research, which connects the 9 way D-socket to a 25 way D-plug (25 way D-sockets are common on RS232 peripherals). For details of how to obtain this cable, see the software and peripherals catalogue included with the ZX Interface 1. This cable is wired as follows:

2. TX data
3. RX data
5. CTS
6. +9v (normally DSR)
7. Ground
20. DTR

Reports

Now that you have attached a ZX Interface 1 to your computer, your programs may produce reports which are not described in Appendix B of the BASIC programming manual. Such reports will be followed by the line number and statement number at which the program stopped.

These new reports are explained (in alphabetical order) below.

Code error

You have tried to **LOAD** a code block that is larger than the destination area specified by your **LOAD** statement.

Drive 'write' protected

You have tried to write data to a Microdrive containing a protected cartridge (i.e. one with the plastic tab on the side removed).

File not found

This means that *either* you have tried to **LOAD** from a file which does not exist, *or* part of the file cannot be found. (This would either be because the file has not been closed, or because the file has been damaged by the power being switched on or off while the cartridge was in a Microdrive.)

Invalid device expression

A device has been specified other than **s,p,k,m,n,t** or **b**. The same report can be produced if a semi-colon, rather than a comma, is used with one of the channels **s,p** or **k**.

Invalid drive number

A Microdrive number has been specified outside the range **1** to **8**.

Invalid name

A file name has been specified as an empty string or a string with more than ten characters.

Invalid station number

A network station has been specified outside the range **0** to **64** (or outside the range **1** to **64** for **FORMAT** statements).

Invalid stream number

A stream has been specified outside the range **0** to **15**.

Merge error

You have tried to **MERGE** data or code. You can only **MERGE** programs. This report will also appear if you have tried to **MERGE** a program saved by the **SAVE . . . LINE . . .** statement.

Microdrive full

You have tried to write data to a cartridge containing no free space. You should therefore run your program or command again, with a cartridge that does contain free space. This can be done by erasing old files from the present cartridge. A file opened for writing on the full cartridge cannot be closed. It should be erased. This will, however, take about thirty seconds because the computer checks the cartridge several times to make sure that the file has no end.

Microdrive not present

You have either tried to use a Microdrive which is not attached to your computer, or a Microdrive which does not contain a cartridge, or a Microdrive containing an unformatted cartridge.

Missing baud rate

The baud rate has not been specified.

Missing drive number

The Microdrive number has not been specified.

Missing name

The file name has not been specified.

Missing station number

The network station number has not been specified.

Program finished

You have tried to execute a line beyond any existing line. This report will appear if a **GOTO** instruction is followed by a number beyond any existing line. It will also appear if **RUN** is typed without a program.

Reading a 'write' file

You have tried to input data from a file which does not yet exist, or which has already been opened for input.

Stream already open

You have tried to **OPEN** to a stream which has already been opened to a new channel (**#n,t** or **b**). The stream can only be opened if it is first closed.

Verification has failed

A saved file does not agree with the program, data or code currently in the computer.

Writing to a 'read' file

You have tried to output data to an existing file. The existing file should be erased if it is not needed. Otherwise, a new file should be used.

Wrong file type

You have tried either to **INPUT** or **MOVE** a saved file *or* to **LOAD, VERIFY** or **MERGE** a **PRINT** type of file, *or* to **LOAD** a **CODE** or **DATA** file as a program (or vice versa).

If you are using **INPUT** then you should be using **LOAD.** If you are using **LOAD** then either you should use the **CODE** or **DATA** options or you should use **INPUT.**

The extended BASIC

The ZX Interface 1 extends the BASIC already in the Spectrum. The extensions and additions are summarised below.

Streams

Streams are specified as #n where n is a number in the range 1—15. Streams 1, 2 and 3 are usually used by BASIC. The # character is part of the keyword for the **OPEN #** and **CLOSE #** statements.

Channels

There are seven types of channel in the extended BASIC; the keyboard (k), the screen (s), the ZX Printer (p), the text RS232 Interface (t), the binary RS232 Interface (b), the network (n) and the Microdrive (m).

Each channel type is specified by its letter which may be upper case or lower case. The network and Microdrive require additional information to specify the channel completely.

A network channel requires a station number, so a network channel is specified as **"n"**;x where x is a station number in the range 0—64.

A Microdrive channel requires a Microdrive number and a file name, so a Microdrive channel is specified as **"m"**;y;**"name"** where y is the Microdrive number in the range of 1—8 and **"name"** is a string of between 1 and 10 characters.

Statements

CAT y	Gives a list of all the files in the cartridge in Microdrive y. The list is presented in alphabetical order and is preceded by the name of the cartridge and followed by the remaining capacity in kilobytes.
CAT # z;y	Sends the catalogue of the cartridge in Microdrive y, as described above, to stream z.
CLOSE # stream	Unlinks any channel from the specified stream. If there is any buffered data then this is either transmitted (on the network) or recorded (on the Microdrive).

Statements

ERASE "m";y;"name"

Erases the file with the specified name from the cartridge in Microdrive y.

FORMAT "m";y;"name"

Prepares a blank Microdrive cartridge in Microdrive y for use by BASIC. The name **"name"** is given to the cartridge and this will appear in catalogues.

FORMAT "n";x

Sets the network station number to x.

FORMAT "t";x
FORMAT "b";x

Set the baud rate for the RS232 Interface to x (x should be chosen from one of the standard baud rates **50, 110, 300, 600, 1200, 2400, 4800, 9600, 19200**).

INKEY$ #Stream

Returns a single character as a string if a character is available, and returns the null string " " if no character is available from the stream.

This instruction is only meaningful if the stream is linked to the network, or the RS232 Interface.

INPUT #Stream;variable

Inputs the variable from the specified stream. The stream must previously have been opened to an input channel. It is important to note that *any print items in the* ***INPUT*** *statement will be* output to the stream. This is usually only required when inputting from the keyboard. It should also be noted that the "," separator outputs a character.

The **LINE** option is available as before.

LOAD * channel options

Loads the program, data or code from the specified channel. Only the channels **"b", "n"** or **"m"** may be used.

All the options available with **LOAD** are available with **LOAD***.

MERGE * channel options

The same as **LOAD** above except that it does not delete old program lines or variables except to make way for new ones with the same line number or name.

Statements

MOVE source TO destination

Moves data from the source to the destination. The source and destination may be either stream numbers or channels.

The command only terminates when an end of file marker is encountered in the source: this can only happen if the source is either a network or Microdrive channel or else a stream linked to one.

If the source or destination is a channel then it is effectively opened first and closed afterwards.

OPEN # stream, channel

Links the specified channel to the specified stream in order to allow BASIC input or output to that channel. The stream must previously be closed or opened to **k**, **s** or **p**.

PRINT # stream . . .

Outputs the print sequence (. . .) to the specified stream. The stream must previously have been opened to an output channel.

The print sequence has the same syntax as before, and may contain further # tokens.

SAVE * channel options

Saves the program, data or code to the specified channel. Only the channels **"b"**, **"n"** or **"m"** may be used.

All the options available with **SAVE** are available with **SAVE ***.

VERIFY * channel options

The same as **LOAD** above except that data is not loaded but is compared with the data already there.

Index

This index includes the keys on the keyboard and how to obtain them.

Sinclair Research Limited
25 Willis Road,
Cambridge CB1 2AQ
England